COMPUTING

VIDEOTECA

manuale 2

Per i possessori di computer SINCLAIR ZX SPECTRUM

PSEUDOCODICE: 3ª lezione

Come sviluppare un programma

Variabili interne e gestione del video

I colori dello Spectrum rifulgono di luce sempre più viva e intensa, l'entusiasmo dei lettori regala energia che sprizza da tutti i computer dell'Editoriale Video.
Lettere e telefonate, giunte a migliaia, ci hanno portato entusiasmo, idee e nuovi programmi. Ed hanno dato un senso e sollievo alle nostre fatiche.
Grazie ai lettori, quindi, e grazie ancora una volta alla piccola e magica scatoletta nera, allo ZX Spectrum che non tradisce entusiasmo e passione di tutti i suoi estimatori.
COMPUTING VIDEOTECA si impreziosisce quindi sempre più, grazie al dialogo coi lettori e diventa ben più che un manuale per conoscere meglio il proprio ZX Spectrum, ricco com'è di segreti, astuzie e regole fondamentali della programmazione. COMPUTING VIDEOTECA vuol diventare uno strumento importante della fantasia e della personalità di coloro che apprezzano il proprio minicomputer.
La raccolta di manuali e cassette diventa sempre più significativa, fino a costituire con lo ZX Spectrum un sistema completo e sempre più potente.
Ai primi dieci numeri, che costituiranno già una raccolta completa e prestigiosa, seguiranno altre raccolte che avranno lo scopo di accompagnare i lettori nell'evoluzione della scienza informatica con il conforto di un linguaggio familiare e di programmi di software realmente funzionali.
Il secondo numero di COMPUTING VIDEOTECA esce in ritardo a causa delle vacanze estive: non potevamo permettere che la chiusura di parte delle edicole creasse disagio alle decine di migliaia di lettori. Ci scusiamo e cercheremo di farci perdonare presto affrendovi bellissime novità.
Arrivederci fra un mese.

PSEUDOCODICE E PROGRAMMAZIONE BASIC

Con l'articolo di questo mese, il terzo della serie, diamo una prima conclusione al discorso sulle "strutture dati interne".

Per strutture dati interne intendiamo tutti i modi di memorizzazione di valori (numerici ed alfanumerici) all'interno di un programma (BASIC).

Le strutture dati interne elementari sono:

— le VARIABILI (vedi articolo numero 1)
— le VARIABILI con INDICE (vedi articolo numero 2)
— le VARIABILI A DUE INDICI (che trattiamo in questo articolo).

Oltre a queste strutture elementari, ne aggiungeremo poi di "complesse", cioè create a partire da queste 3 elementari, e a ciò dedicheremo un articolo più avanti.

Il modo più semplice per comprendere le variabili a doppio indice è pensare alla battaglia navale. In quel gioco, ogni quadretto è individuato da una lettera ed un numero: per esempio A8, oppure C15. Proviamo a pensare che invece delle lettere si usino numeri in entrambe le direzioni: invece di A8 dovremo scrivere qualcosa come: (1,8), ed invece di C15 scriveremo (3,5). Osservate che dobbiamo mettere la virgola per separare correttamente i numeri, e racchiudiamo tra parentesi la coppia di valori che individuano il quadretto.

A questo punto siamo in grado di visualizzare le variabili a doppio indice: pensiamo che ogni quadretto sia una variabile semplice (che può contenere un valore numerico od alfanumerico) ed il gioco è fatto: quando scriviamo A(1,8) individuiamo la variabile che si trova nella prima riga ed all'ottavo posto della nostra struttura. In figura 1 si vede come si collocano schematicamente le variabili A(X,Y) di una struttura a 4 righe e 5 colonne.

Una struttura di questo tipo si chiama MATRICE, così come la struttura ad un indice si chiamava VETTORE. Il numero di indici di una struttura elementare si chiama DIMENSIONE. Con questa terminologia possiamo sinteticamente riassumere il tutto dicendo:

— una VARIABILE semplice è una struttura senza DIMENSIONE
— una VARIABILE con indice appartiene ad una struttura ad 1 DIMENSIONE, chiamata VETTORE
— una variabile a doppio indice appartiene ad una struttura a 2 DIMENSIONI, detta MATRICE.

Le matrici sono fondamentali, sia in BASIC che per lo pseudo-

codice. La maggior parte degli algoritmi di un certo grado di complessità utilizzano matrici.
In pseudocodice siamo ovviamente autorizzati a utilizzare variabili a doppio indice in qualunque punto del testo. Analogamente a quanto avviene per le variabili ad un indice, l'uso di una variabile a doppio indice sottointende l'esistenza della matrice. Per esempio, se in pseudocodice leggessimo:

A = B(1) + C(2,1)

siamo autorizzati a concludere che in quel programma esiste un vettore, di nome B, ed una matrice, di nome C.
Naturalmente se al posto dei valori espliciti usiamo dei nomi di variabile, si intende che gli indici delle variabili saranno valutati prima di procedere alla interpretazione dell'istruzione: per esempio l'istruzione sopra è equivalente alle seguenti tre:

1) K = 1
2) J = 2
3) A = B(K) + C(J,K)

Insomma, il discorso è analogo a quello dei vettori, visto il mese scorso, con l'unica differenza che nelle matrici ci troviamo un indice in più.
Ovviamente, in BASIC, siamo tenuti a dichiarare le strutture dati: per dichiarare le matrici si usa l'istruzione DIM seguita dal nome della struttura, con indicazione delle righe e colonne della matrice. Per esempio DIM C(4,8) significa che C è una matrice di 4 righe per 8 colonne.

LA TABELLINA PITAGORICA

Come al solito, facciamo un esempio pratico di uso delle variabili a doppi indici.
Supponiamo di voler costruire un programma che faccia comparire sul video la tabellina pitagorica.
Cominciamo osservando che ciascun numero della tabellina pitagorica è il risultato della moltiplicazione del numero di riga per il numero di colonna al cui incrocio quel numero è situato.
Osservate che parlando di matrici è comodo parlare in termini di righe e colonne: inoltre è facile vedere che le variabili sulla stessa riga hanno uguale il valore del primo indice, mentre le variabili sulla stessa colonna hanno ugual valore del secondo indice (vedi fig. 1).
Per "spazzolare una riga" di una matrice, ossia per generare una scansione di una riga, si usa tener fisso l'indice di riga e costruire una ripetizione enumera-

tiva del valore dell'indice di colonna. Per esempio, per spazzolare la riga del numero 3 della tabellina pitagorica potremmo scrivere:

```
INIZIO. Spazzolamento riga
        del 3
 RIPETI PER J. CHE VARIA
 DA 1 A 10
 Scrivi 3 * J.
 FINE_RIPETI
FINE
```

In BASIC:

```
FOR J. = 1 TO 10
 PRINT 3 * J.;
NEXT I
```

Ora guardate: se volessimo spazzolare l'intera matrice, potremmo scrivere dieci volte quelle 3 istruzioni, modificando il 3 in 1,2,3,...fino a 10. Ma non è proprio un caso in cui conviene usare una variabile? Senz'altro. Perciò modifichiamo il nostro pseudocodice così:

```
INIZIO. Spazzolamento della
        matrice.
 RIPETI
 PER I DA 1 A 10
   RIPETI
   PER J DA 1 A 10
     Scrivi I * J.
   FINE_RIPETI
 FINE_ RIPETI
FINE.
```

Abbiamo scritto un rapido algoritmo che scrive tutti i numeri della tabellina pitagorica, uno di seguito all'altro. Abbiamo usato una figura di programmazione molto importante: 2 ripetizioni una dentro all'altra. In inglese si dice che abbiamo fatto usodi un NESTED LOOP, che tradotto letteralmente diventa "ciclo ANNIDATO". La parola NESTED ha in inglese proprio il significato di "una cosa dentro all'altra", come per esempio le scatole cinesi.

Il CICLO ANNIDATO è la figura base, usatissima per spazzolare matrici.

In fig. 2 vedete la versione BASIC dell'algoritmo, dove è stata aggiunta una istruzione di PRINT senza parametri, al termine del ciclo più interno. Questa istruzione serve per far andare a capo il BASIC, dopo aver scritto una riga della tabellina. Vi consiglio di provare subito il programma con il vostro personal. Provate a farlo girare con e senza l'istruzione PRINT tra il primo ed il secondo NEXT. Visto la differenza? (Badate però di mettere il punto e virgola al fondo della prima istruzione PRINT: questo serve per far stampare i numeri uno di seguito all'altro finché appartengono tutti alla stessa riga).

Fino ad ora non abbiamo ancora introdotto le nostre variabili a doppio indice. Supponiamo pe-

rò di voler memorizzare tutti i valori della tabellina pitagorica. In tutto sono 100 numeri. Non ha certo senso pensare di memorizzarli in un vettore: la variabile A(28) per esempio, che numero conterebbe? Solo dopo un certo ragionamento ci verrebbe chiaro che deve contenere il numero 24 (cioè 3 × 8!!).

È più semplice pensare che organizziamo i dati in una matrice, dove viene chiaramente evidenziato il numero di riga ed il numero di colonna come entità separate: allora avremmo che la variabile A(3,8) contiene il dato all'incrocio tra la terza riga e la quarta colonna, cioè proprio 3 × 8.

Scriviamo lo pseudocodice di un algoritmo che memorizzi questa tabellina in una matrice:

```
INIZIO. Memorizzazione
        della matrice.
 RIPETI
 PER I DA 1 A 10
   RIPETI
   PER J DA 1 A 10
     A(I,J) = I * J
   FINE_RIPETI
 FINE_ RIPETI
FINE.
```

Al termine del programma, avremo memorizzato nella matrice A i cento numeri della tabellina pitagorica. Osservate il gioco delle variabili I e J, che sono usate come variabili numeriche per il calcolo, e come indici per individuare la variabile a doppio indice.

Vi consiglio di provare questo programma in BASIC (figura 3), e al termine del RUN di provare a chiedere i valori di variabili a caso tra quelle della matrice: scrivendo cose tipo PRINT A(3,5), oppure PRINT A(4,6)/2, eccetera.

Con questo programma otteniamo la memorizzazione della matrice ma non vediamo nulla sul video. Aggiungiamo ora delle istruzioni che servono per fare il DISPLAY (cioè la visualizzazione) della tabella.

Anche in questo caso si tratta di spazzolare la matrice e stamparne i valori in modo che siano ordinati per righe:

```
INIZIO. Display di una
        matrice.
 RIPETI
   PER I DA 1 A 10
     RIPETI
     PER J DA 1 A 10
      Scrivi A(I,J) di seguito
     FINE_RIPETI
     Salta a capo
 FINE_RIPETI
FINE
```

Con questo algoritmo, variando opportunamente i limiti di ripetizione (cioè cambiando i 10 in qualche altro valore), avete la

possibilità di fare il display di una qualunque matrice.
In figura 4 vedete un programma BASIC che prima carica la matrice A con i valori della tabellina pitagorica (istruzioni 15-50); e poi stampa la tabellina, racchiudendola in una cornicetta.
Le istruzioni PRINT necessitano di un minimo di spiegazione.
L'istruzione 5 e 10 stampano il titolo e la riga orizzontale in alto del riquadro in cui è posta la tabellina.
L'istruzione 100 scrive l'elemento A(I,J) della tabella saltando 4 posizioni dopo ogni numero stampato per 10 volte. Questo assicura che i dati siano incolonnati e adeguatamente separati.
Badate che ora il programma si può considerare come diviso in sezioni logiche:
— caricamento dei valori della matrice
— stampa della matrice.
Le due sezioni hanno la struttura di un algoritmo completo, e svolgono due funzioni generalizzate. (Vedremo nel prossimo articolo che questo aspetto si può sfruttare per definire dei sottoprogrammi).

LE MATRICI NEI GIOCHI

È difficile sopravvalutare l'importanza delle matrici negli algoritmi che realizzano giochi. Diamo qui alcuni esempi.

— è una matrice l'insieme dei punti che individuano un elemento (figura) in un disegno animato

— è una matrice la struttura che in molti giochi tiene traccia dei movimenti del giocatore, e che controlla la scelta di possibilità di mosse di un giocatore in un certo punto di un gioco (come per esempio in ADVENTURE)

— o sono le matrici le strutture base per la numorizzazione di tabelle, liste, alberi ed altre strutture complesse che guidano le scelte nei giochi più sofisticati.

Facciamo ora un esempio pratico molto semplice. Pensiamo alla BATTAGLIA NAVALE. Pur senza pretendere di voler creare un gioco completo, proviamo a pensare come potremmo realizzarlo.
Subito ci viene in mente che potremmo far uso di una matrice, supponiamo M, di grandezza adeguata per memorizzare tutti i punti del nostro specchio di mare.

Supponiamo di giocare su un quadrato 20 × 20. Avremo bisogno di una matrice M(20,20). Inizialmente questa matrice contiene tutti numeri zero. Decidiamo perciò che se M(x,y) contiene 0 significa che nel punto (x, y) non vi è altro che "mare".
Decidiamo allora di posizionare delle navi. Questo posizionamento deve essere fatto automaticamente dal Personal senza il nostro interessamento, altrimenti che gioco sarebbe, se sapessimo dove il PC ha messo le navi?
Per questo usiamo una funzione base del BASIC: l'estrazione di un NUMERO CASUALE. Con questa istruzione (che è un po' diversa da BASIC a BASIC) otteniamo l'effetto di una estrazione di un numero a caso tra due estremi che possiamo fissare noi.
Per il nostro esempio supponiamo di estrarre una coppia di numeri a caso compresi tra 1 e 20. Otteniamo così un punto a caso di coordinate X, Y. Supponiamo di avere estratto (5,10): in questo caso la nostra nave avrà come punto di partenza il punto (5,10).
[Ora abbiamo ancora da decidere se piazzare la nave in verticale o in orizzontale (potremmo estrarre un nuovo numero casuale tra 0 ed 1), e poi stare attenti a non uscire dai bordi].
Non entriamo nei dettagli: la sostanza è che una volta individuati i punti che intendiamo assegnare alla nave possiamo pensare di METTERE un 1 NEGLI ELEMENTI corrispondenti della matrice. Se la nave, per esempio, fosse lunga 3 caselle in orizzontale, e partisse da (5,10), allora dovremmo assegnare 1 ai seguenti elementi:

M(5,10); M(6,10); M(7,10).

Ora basta costruire il programma di input delle coordinate in cui vogliamo sparare e leggere i due valori nelle variabili, P1 e P2, per esempio.
Osservate ora cosa possiamo fare:

— se M(P1,P2) contiene 0 abbiamo fatto "acqua"

— se M(P1,P2) contiene 1 abbiamo colpito una nave.

Per evitare colpi ripetuti possiamo pensare che dopo un colpo su un elemento che contiene 0 oppure 1, SOMMIAMO 2 a quell'elemento. A questo punto abbiamo a disposizione i seguenti codici:

Contenuto dell'elemento M(P1,P2)	Significato
0	Mare
1	Nave non colpita
2	Mare. Colpo già sparato
3	Nave colpita

In uno degli articoli seguenti vedremo qualche altro particolare realizzativo di questo semplice gioco.
Magari, nel frattempo, potreste provare qualcosa da soli. Buon divertimento ed Arrivederci.

- 3 - continua

M.S.

A(1,1)	A(1,2	A(1,3)	A(1,4)	A(1,5)
A(2,1)	A(2,2)	A(2,3)	A(2,4)	A(2,5)
A(3,1)	A(3,2)	A(3,3)	A(3,4)	A(3,5)
A(4,1)	A(4,2)	A(4,3)	A(4,4)	A(4,5)

Fig. 1. Ecco come si collocano visivamente le variabili di una struttura a doppio indice.

```
FOR I = 10 TO 10
   FOR J = 1 TO 10
      PRINT I * J;
   NEXT J
        PRINT
NEXT I
```

Fig. 2. Il più semplice programma per generare la tabellina pitagorica.

```
DIM A(10,10)
FOR I = 1 TO 10
   FOR J = 1 TO 10
      A(I,J) = I * J
   NEXT J
NEXT I
```

Fig. 3. Programma per la numorizzazione della tabellina in una matrice.

```
1 DIM A(10,10):PRINTCHR$(147)
5 PRINT:PRINT:PRINT "                    TABELLA PITAGORICA"
10 PRINT "----------------------------------------"
15 FOR I = 1 TO 10
20   FOR J = 1 TO 10
30     A(I,J)=I*J
40   NEXT J
50 NEXT I
60 REM .. SCRIVIAMO LA TABELLA
70  FOR I = 1 TO 10
90   FOR J = 1 TO 10
100    PRINT TAB(J*4-4) A(I,J);
110   NEXT J
130 NEXT I
140 PRINT:PRINT "----------------------------------------"

READY.
```

Fig. 4. Programma per generare la tabellina pitagorica.

```
                     tabellina pitagorica
+--------------------------------------------------------------+
I     1    2    3    4    5    6    7    8    9   10           I
I     2    4    6    8   10   12   14   16   18   20           I
I     3    6    9   12   15   18   21   24   27   30           I
I     4    8   12   16   20   24   28   32   36   40           I
I     5   10   15   20   25   30   35   40   45   50           I
I     6   12   18   24   30   36   42   48   54   60           I
I     7   14   21   28   35   42   49   56   63   70           I
I     8   16   24   32   40   48   56   64   72   80           I
I     9   18   27   36   45   54   63   72   81   90           I
I    10   20   30   40   50   60   70   80   90  100           I
+--------------------------------------------------------------+
```

Fig. 5. Il risultato del programma di figura 4.

COME SVILUPPARE UN PROGRAMMA

In queste pagine cercheremo di introdurre nel modo più chiaro possibile le tappe da seguire per un corretto sviluppo dei programmi. Benché i concetti illustrati si riferiscano in particolare alla tecniche di creazione dei giochi, il discorso risulta valido in generale e sarebbe buona prassi seguire sempre strade di questo genere per lo sviluppo di qualsiasi programma.
Questa esposizione sarà la più generale possibile, poiché, qualunque sia la natura di un programma, la sua realizzazione richiede il passaggio per un certo numero di fasi ben definite che di solito vengono raggruppate in cinque blocchi che sono, nell'ordine:
— Le idee di partenza, l'immaginazione, l'esposizione del problema.
— Il quaderno degli appunti, l'insieme delle funzioni da realizzare.
— Il diagramma di flusso, l'algoritmo di risoluzione.
— La traduzione in un linguaggio di programmazione evoluto.
— La messa a punto, le prove di funzionamento.
Questo metodo di programmazione così definito permette d'evitare il tradizionale errore consistente nello scrivere un programma direttamente, senza preliminari. Questo sistema non è certo razionale e non può che allungare enormemente i tempi di messa a punto. Se avrete comunque la fortuna di riuscire a far girare il vostro programma nonostante tutto, esso si presenterà sotto forma di aggiustamenti e manipolazioni successive, sarà quindi molto difficile se non impossibile interpretarlo o modificarlo in un secondo tempo.
Andiamo dunque a esaminare queste tappe che costituiscono IL metodo per realizzare un programma corretto.

1 — LE IDEE

Il lavoro di immaginazione costituisce la prima fase nella concezione di un gioco. In un primo tempo, tutte le vostre idee, anche le più strampalate, possono essere all'origine di giochi molto personali, che vi divertiranno sicuramente più di quelli "classici" che potete trovare ovunque.
Tuttavia, prima di lanciarvi nelle tappe successive, occorre che siate certi che il progetto che avete in mente sia realizzabile. In effetti, nonostante tutta la libertà di cui disponete, ci sono comunque alcune costrizioni che possono o meno trasformare in un vero gioco le vostre

idee di partenza.
Ma cos'è un "vero gioco?" Ecco alcuni criteri da tenere presenti per fare in modo che il gioco sia il più interessante possibile.
Il gioco deve mettere alla prova differenti qualità di un giocatore:
— la sua intelligenza nei giochi di riflessione,
— i suoi riflessi nei giochi d'azione,
— la sua precisione nei giochi di abilità,
— la sua fortuna nei giochi d'azzardo,
deve inoltre fare in modo che il giocatore partecipi attivamente, per mezzo della tastiera o del joystick.
Un gioco deve inoltre essere equilibrato, cioè nè troppo facile nè troppo complicato: in effetti un gioco al quale si vince o si perde sempre non presenta alcun interesse, e non lo utilizzereste mai. Occorre dunque saper dosare opportunamente la difficoltà di un gioco perché esso attiri la vostra attenzione il più a lungo possibile, senza annoiarvi.
ESEMPIO:
Supponiamo che vogliate realizzare un gioco di sci. Dovrete visualizzare uno sciatore, le bandierine entro cui deve passare e alcuni alberi che fungeranno da ostacoli naturali. Questo gioco risponde abbastanza bene ai requisiti che abbiamo appena esposto. Occorre infatti cambiare la direzione dello sciatore in funzione dei tasti premuti, controllare se colpisce un albero, ecc.
Ci occuperemo comunque dello sviluppo di questo gioco nelle tappe successive.

2 — IL QUADERNO DEGLI APPUNTI

Il quaderno degli appunti costituisce in qualche modo il supporto della vostra immaginazione. Esso è indispensabile per descrivere, a partire dalle vostre idee originali, quello che il programma deve fare, i suoi obiettivi, e nel caso di un gioco, le sue regole, la presentazione, il ruolo del giocatore e quello della macchina, il modo in cui va effettuato il dialogo uomo-computer...
Tutti questi punti importanti costituiscono la base per le tappe successive e devono essere scrupolosamente rispettati. Vedremo nelle tappe che seguono come gli errori commessi sul quaderno degli appunti possano essere la causa di ritardi imprevisti nei tempi di esecuzione, difficoltà algoritmiche, e altro...

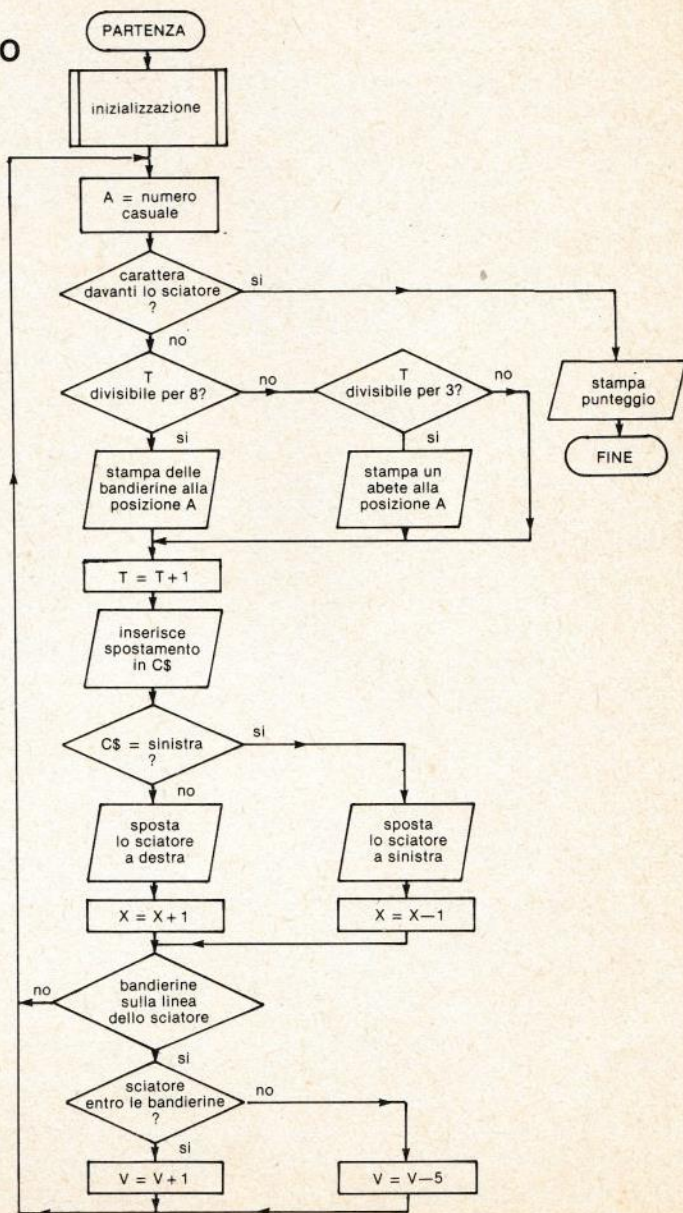
ESEMPIO:
Continuiamo lo sviluppo del nostro gioco di sci, e stabiliamo le sue regole sul quaderno degli appunti.
— Il gioco consiste nel guidare uno sciatore in uno slalom, facendolo passare fra le bandierine.
— Lo sciatore deve scendere verticalmente dall'alto dello schermo.
— Passando fra le bandierine viene incrementato di 1 il punteggio.
— Se lo sciatore sorpassa le bandierine senza passarvi in mezzo vengono sottratti 5 punti al totale.
— Se lo sciatore urta un albero, viene mostrato il punteggio e la partita termina.

3 — IL DIAGRAMMA DI FLUSSO

È in questa fase della programmazione che voi inizierete a "creare" a grandi linee la struttura del programma, basandovi su quanto avete scritto nella parte concernente il quaderno degli appunti. Questo organigramma si potrà presentare sotto forma di un disegno (diagramma di flusso), o di una serie di istruzioni strutturate in modo da fare apparire chiaramente la costituzione del programma (pseudocodice).
In questo caso useremo il diagramma di flusso per rappresentare le funzioni svolte dal programma, il discorso è comunque applicabile anche nel caso dello pseudocodice.
Bisogna innanzitutto definire i blocchi che devono corrispondere a delle funzioni elementari del programma. Questa struttura a blocchi distinti permette una migliore comprensione e la messa a punto di una determinata parte indipendentemente dalle altre, il che è sicuramente molto efficace per rintracciare eventuali errori.
La suddivisione in blocchi non corrisponde necessariamente all'utilizzazione di sottoprogrammi, ma soprattutto a una separazione delle differenti unità logiche dell'insieme. Tuttavia bisogna tener presente che sovente queste strutture faranno uso di sottoprogrammi come potete facilmente constatare dai giochi che seguono l'articolo.
L'operazione di strutturazione di un programma può, malgrado tutto, rivelarsi difficile, se non proprio quasi impossibile, per ragioni già menzionate nel punto riguardante il quaderno degli appunti: eccessiva complessità dell'algoritmo, lunghezza del programma (in certi casi la ca-

pacità di memoria può essere un fattore critico), ecc. Per questi motivi sarete spesso costretti a rivedere e modificare certi termini che avevate precedentemente stabilito sul quaderno degli appunti, in modo da rendere realizzabile il vostro programma. Vediamo dunque come si può disegnare un diagramma di flusso.

ESEMPIO:
Vediamo come si può strutturare il diagramma di flusso del gioco dello sci che ci siamo proposti di realizzare.
L'organigramma generale di un programma comporta spesso l'uso di vari sotto-organigrammi, corrispondenti ciascuno a un blocco logico del gioco. In questo esempio abbiamo disegnato il diagramma di flusso del programma principale del nostro gioco dello sci.
Nel blocco INIZIALIZZAZIONE, dovranno trovarsi i valori iniziali delle variabili, le definizioni dei caratteri grafici, il colore di sfondo e degli oggetti, ecc. La variabile T contiene il numero di linee già passate e determina, in funzione di queste ultime, la visualizzazione di un albero o delle bandierine, la cui posizione di stampa è casuale e dipende dalla variabile A.

DIAGRAMMA DI FLUSSO DELLO SCI

4 — LA CODIFICA

Quando il diagramma di flusso è stato disegnato, l'algoritmo stabilito in modo preciso, e quando la logica di ogni blocco costituente il programma è stata provata, è possibile passare alla traduzione in un linguaggio di programmazione evoluto, che nel nostro caso sarà il BASIC.
A partire da questo punto sono indispensabili conoscenze specifiche. Bisogna conoscere infatti le parole chiave e la sintassi del linguaggio di programmazione che avete deciso di usare, per poter tradurre senza difficoltà l'algoritmo da voi sviluppato. Tenete presente che le tappe precedenti generalmente non facevano appello a conoscenze specifiche di informatica, e tutto ciò che occorreva era un po' di buon senso.
Per procedere alla codifica è bene seguire un procedimento razionale. Per esempio, nella numerazione delle linee di programma è consigliabile procedere di 10 in 10, allo scopo di poter inserire altre linee durante la messa a punto, senza per questo spostare interi blocchi di programma, il che oltre a essere estremamente fastidioso è spesso causa di nuovi errori. Conviene anche numerare ogni sotto-programma in modo che si distacchi il più possibile dagli altri, per poter facilitare l'eventuale rilocatura del programma nel corso delle modifiche, potreste ad esempio disporle di 1000 in 1000.

ESEMPIO:

Avrete modo di osservare quanto scritto in questa fase guardando il gioco di sci riportato più avanti.

5 — LA MESSA A PUNTO

In teoria questa fase è superflua, e il programma dovrebbe poter funzionare senza passare per la messa a punto. Malauguratamente, un programma contiene quasi sempre alcuni errori che possono essere di diversi tipi.
Gli errori algoritmici sono i più gravi. In questi casi si è costretti a rifare le due fasi precedenti o addirittura anche la prima. In linea di massima, questo tipo di errore è dovuto a una scorretta analisi del problema, e un programmatore attento non dovrebbe trovarsi di fronte a questi problemi.
Gli errori comuni si situano di solito nel passaggio dal diagramma di flusso al programma. Il miglior sistema per correggerli consiste nel controlla-

re l'esecuzione del programma blocco per blocco, sarà quindi nel vostro interesse aver strutturato il programma in maniera modulare e aver adottato una numerazione razionale.

6 — CONCLUSIONI

Speriamo di aver chiarito in queste pagine le modalità per un corretto sviluppo dei vostri programmi, siano essi giochi o altro.

Anche se a prima vista il metodo esposto può sembrare lungo e complesso, esso è in realtà il sistema più semplice e razionale per creare degli ottimi programmi, e siamo certi che seguendo i nostri consigli diverrete ben presto abilissimi programmatori.

Massimo Cellini

PROGRAMMA DELLO SCI

```
  10 REM © M. Cellini
  15 BORDER 1: PAPER 1: INK 6: B
RIGHT 1: CLS : PRINT AT 10,10;"S
UPERSLALOM";AT 21,8; FLASH 1;"pr
emi un tasto"
  20 GO SUB 180
  25 PAUSE 0
  30 BORDER 7: PAPER 7: INK 0: B
RIGHT 0: CLS
  35 LET x=16: LET t=0: LET c=11
2: LET v=0
  40 REM INIZIO
  45 LET a=INT (RND*15)+7: POKE
23692,255
  50 IF POINT (x*8,159)=1 OR POI
NT ((x+1)*8,159)=1 OR POINT ((x+
2)*8,159)=1 THEN GO TO 145
  55 IF t/8<>INT (t/8) THEN GO T
O 75
  60 PRINT AT 21,a; INK 2;"▲
▲"
  65 PRINT INK 7;CHR$ (a+40);TAB
 a; INK 2;"|        |"
  70 PRINT : GO TO 95
  75 IF t/3<>(t/3) THEN GO TO 90
```

```
  80 PRINT AT 21,a; INK 4;"▲":
PRINT TAB a; INK 4;"♣"
  85 PRINT : GO TO 95
  90 PRINT AT 21,0: PRINT : PRIN
T
  95 LET t=t+1: PRINT AT 0,29;v
 100 LET c$=INKEY$: IF c$<>"" TH
EN LET c=CODE c$
 105 IF c=113 THEN PRINT AT 0,x;
"♠": PRINT AT 1,x;"⅄": IF x>6
THEN LET x=x-1
 110 IF c=112 THEN PRINT AT 0,x;
"♠": PRINT AT 1,x;"⅃": IF x<28
 THEN LET x=x+1
 115 PAUSE 5
 120 LET p=CODE SCREEN$ (2,0): I
F p=32 THEN GO TO 45
 125 IF x>p-41 AND x<p-33 THEN L
ET v=v+1: GO TO 45
 130 LET v=v-5
 135 GO TO 45
 140 REM FINE
 145 FOR i=-50 TO 50
 150 BEEP .01,i
 155 NEXT i
 160 BORDER 0: PAPER 1: INK 7: C
LS : PRINT AT 10,2;"PREMI UN TAS
TO PER RIGIOCARE"
 165 IF INKEY$="" THEN GO TO 165
 170 GO TO 30
 175 REM CAR. GRAFICI
 180 DATA 3,3,1,7,15,11,19,11,12
8,128,0,192,224,160,192,160
 185 DATA 6,3,2,2,3,2,21,10,148,
168,212,160,192,128,0,0
 190 DATA 1,1,0,3,7,5,3,5,192,19
2,128,224,240,208,200,208
 195 DATA 41,21,43,5,3,1,0,0,96,
192,64,64,192,64,168,80
 200 DATA 0,0,1,3,7,15,15,31,0,0
,0,128,192,224,224,240
 205 DATA 63,63,127,255,3,3,3,0,
248,248,252,254,128,128,128,0
 210 DATA 1,3,7,15,31,63,1,1,1,1
,1,1,1,1,1,1
 215 FOR i=0 TO 111
 220 READ a: POKE USR "a"+i,a
 225 NEXT i
 230 RETURN
```

VARIABILI INTERNE E GESTIONE DEL VIDEO

Fino a pochi anni fa, i computer facevano bella mostra di sè nelle grandi agenzie, nelle banche, o al più in qualche avveniristico ufficio. Ciò è comprensibile, dato che il costo medio di un piccolo computer superava abbondantemente le disponibilità economiche di un privato che ne volesse fare un uso "domestico", per non parlare poi dei giovani appassionati che sono tradizionalmente i più squattrinati.
Improvvisamente, in un memorabile giorno di alcuni anni addietro (la data esatta si è ormai persa nella leggenda), ecco apparire un computer, un vero computer, ma dalle dimensioni tanto ridotte da poter essere scambiato con il suo manuale! Quel computer era l'ormai dimenticato (non da quelli che hanno avuto la fortuna di possederlo) SINCLAIR ZX 80, un computer che passerà alla storia per aver rivoluzionato il mondo dell'informatica, permettendo a quanti lo desideravano di acquistare un computer da tenere in casa, per programmare, giocare, tenere la contabilità, ecc.
Dopo circa un anno dalla presentazione dello ZX 80, la Sinclair inizia a commercializzare un altro straordinario computer: lo ZX 81, una macchina ancora più piccola del suo predecessore, ma decisamente migliorata, senza per questo intaccare il prezzo, che anzi viene ulteriormente abbassato. Questo innovativo computer (ancora in produzione) ha dato un impulso decisivo all'affermazione dell'informatica domestica in Italia e nel mondo.
Ma non finisce qui. Non contenta, la Sinclair produce un terzo computer veramente rivoluzionario, che in breve tempo ha surclassato una marea di concorrenti grazie alle sue ineguagliabili doti unite a una estrema semplicità d'uso che ne fanno la macchina ideale sia per chi voglia inoltrarsi nel mondo del-

la programmazione, sia per chi la voglia usare per applicazioni hobbistiche o semiprofessionali. Questo computer è, come avrete capito, lo ZX SPECTRUM.
Come abbiamo già detto, imparare a programmare sullo Spectrum è veramente un... gioco da ragazzi, ma saperlo padroneggiare pienamente richiede una conoscenza approfondita della macchina e soprattutto molta esperienza. In queste pagine cercheremo comunque di chiarire alcuni concetti espressi poco chiaramente nel manuale che viene dato a corredo della macchina, e vi sveleremo qualche interessante e divertente trucchetto possibile con questo straordinario computer.
Tralasciamo di spiegare i vari comandi BASIC che ormai tutti dovrebbero conoscere, e ci caliamo subito nel cuore dello Spectrum, dove si trovano le cose più interessanti.
La RAM dello Spectrum in realtà non è tutta disponibile per il BASIC, infatti una parte di essa (6912 bytes) è riservata al display file, un'altra parte è riservata al buffer della stampante, un'altra ancora è occupata dai caratteri grafici e, per finire, altri 18Ø bytes sono occupati dalle variabili del sistema.
Vediamo più da vicino cosa contengono queste variabili e cosa si può ottenere alterandole.
Sia nel 16 che nel 48 K, le variabili di sistema iniziano alla locazione 23552 e terminano alla 23733. È inutile soffermarsi a elencarle, poichè a questo provvede già il manuale della Sinclair.
Cerchiamo quindi di approfondire la conoscenza di alcune di esse: la prima variabile interessante che troviamo è la REPDEL, locata all'indirizzo 23561: essa contiene il valore corrispondente al tempo (in cinquantesimi di sec.), per cui è necessario tenere premuto un tasto perchè inizi a ripetersi. Inserendo un valore di Ø si elimina il repeat, mentre con valori molto bassi si ottiene una ripetizione quasi istantanea. Questo può essere molto utile per esempio nei giochi, specialmente se si altera anche la variabile successiva (23562) che controlla il ritardo fra una ripetizione e la successiva di un tasto premuto.
Un'altra variabile molto interessante è la CHARS, che occupa due bytes: 236Ø6 e 236Ø7 (le variabili a 2 bytes possono contenere fino a un valore di 65535, infatti 256 × 256 = 65536). Que-

sta variabile contiene l'indirizzo del set di caratteri standard che normalmente si trova in ROM, ma può essere facilmente ridefinito in RAM, anche se l'operazione è piuttosto lunga; vediamo come si può procedere.
Per prima cosa occorre porre nella variabile il valore corrispondente al nuovo indirizzo del set di caratteri, nell'esempio lo poniamo a partire dalla locazione 3ØØØØ. Bisogna sapere che una variabile a due bytes è formata da un byte basso (il primo) e un byte alto (il secondo), quest'ultimo corrispondente al suo valore moltiplicato per 256, infatti dato che il primo byte non può contenere più di 256, ogni volta che raggiunge questa cifra si azzera e incrementa di una unità il secondo byte (alto). Da quanto detto si può dedurre che per leggere il valore di una variabile a 2 bytes occorre un comando di questo genere:

```
PRINT PEEK x + 256 *
PEEK x + 1
```

che somma al primo byte il valore del secondo byte moltiplicato per 256, dove x è il primo byte (basso) e x + 1 il secondo (alto). Ne consegue che per scrivere un numero in una variabile a due bytes bisogna dividere il numero stesso per 256 e porre il risultato nel byte alto, mentre il resto della divisione va posto nel byte basso. Esiste comunque un semplice trucchetto per evitare questo noioso lavoro che sicuramente sarà bene accetto da quanti nutrono ancora una profonda idiosincrasia per la matematica.
È infatti sufficiente inserire la seguente linea:

```
RANDOMIZE xxx:
PRINT PEEK 2367Ø:
PRINT PEEK 23671
```

in questo modo vi verrà visualizzato prima il valore del byte basso e poi quello del byte alto corrispondenti al numero xxx da voi scritto dopo il comando RANDOMIZE. Ciò è facilmente spiegabile, infatti il comando RANDOMIZE serve a alterare il seme da usare nell'istruzione RND. Questo seme viene memorizzato automaticamente in due bytes delle variabili del sistema che sono appunto la 2376Ø e la 23761; basta quindi andare a leggere queste ultime per ottenere i due numeri corrispondenti alla cifra da voi inserita.
Torniamo ora al nostro set di caratteri. Dopo aver alterato opportunamente il puntatore, occorre inserire i nuovi valori da

porre nei bytes per la definizione dei caratteri, allo stesso modo di come si fa per i caratteri grafici, definendo cioè 8 bytes per ogni carattere, ognuno dei quali corrisponde a una riga del carattere stesso che, come certo ricorderete, è costituito da una matrice di 8 × 8 punti grafici. Il set di caratteri inizia con lo spazio e termina con il simbolo di copyright (CHR$ 32-127).
Vediamo dunque come procedere in pratica alla creazione di un nuovo set di caratteri.

```
1∅   POKE 236∅6,48:
     POKE 236∅7,117
2∅   FOR C = 32 TO 127
3∅   FOR B = ∅ TO 7
35   READ N
4∅   POKE 3∅∅∅∅ +
     + (C*8) + B,N
5∅   NEXT B
6∅   NEXT C
1∅∅  DATA (CHR$ 32)
11∅  DATA (CHR$ 33)
...
```

In ogni linea di DATA dovete porre gli 8 bytes corrispondenti al carattere che volete creare, tenendo conto che il CHR$ 32 è uno spazio, il CHR$ 33 un punto esclamativo, ecc. Terminata l'operazione di inserimento, potete proteggere il vostro nuovo set di caratteri con un CLEAR 29999 (naturalmente se avete usato come locazione di partenza la 3∅∅∅∅), in questo modo anche dando il NEW, avrete comunque in memoria i caratteri ridefiniti. Se siete soddisfatti del vostro lavoro, potete salvare su nastro l'intero set con SAVE "nome" CODE 3∅∅∅∅,1024 e ricaricarlo successivamente con LOAD "nome" CODE 3∅∅∅∅: CLEAR 29999 basterà poi dare i due POKE come riportato nella linea 1∅ del programma per usare il vostro nuovo set.
Se avete ancora dei dubbi in proposito, leggetevi attentamente il capitolo 23 del manuale italiano dello Spectrum, dopodichè rileggete dal principio questo articolo e vedrete che tutto vi apparirà più chiaro.
Proseguendo il nostro viaggio tra le variabili del sistema, troviamo a breve distanza un'altra locazione di controllo di grande utiità. Si tratta della variabile PIP, all'indirizzo 236∅9; essa controlla la lunghezza del clic dei tasti e inizialmente è posta a zero, ma può essere utilmente alterata ponendovi per esempio un valore di 1∅ o 2∅, si ottiene così il risultato di creare un software clic decisamente più udibile di quello standard, il chè permette una battitura più rapida e sicura.

Dopo quest'ultima variabile, ci sono un certo numero di locazioni di controllo e vari puntatori che, per il momento, non prenderemo in considerazione, ma che si riveleranno alquanto utili quando programmerete in modo più sofisticato. Saltiamo quindi alla locazione 23692 dove troviamo la variabile SCRCT, contenente il numero di scroll che devono essere eseguiti prima di fermarsi a porre la domanda "scroll?", quindi, per esempio, se vi ponete il valore 255, verranno eseguiti 254 scroll automatici.

L'ultima variabile interessante che troviamo è quella locata all'indirizzo 2373∅-23731. Anche questa è una variabile a 2 bytes e deve quindi essere letta come spiegato precedentemente. In questi due bytes è immagazzinato l'indirizzo attuale dell'ultimo byte nell'area di lavoro del sistema BASIC, cioè la locazione della RAMPOT in quel momento.

Per adesso terminiamo quì la nostra escursione fra le variabili interne dello Spectrum, ma presto torneremo su questo interessante discorso per svelarvi altri piccoli segreti sul vostro insostituibile amico ZX.

LA GESTIONE DEL VIDEO

Passiamo quindi a un altro argomento molto interessante riguardante la gestione del video. Come ben saprete, lo Spectrum riserva una certa parte della RAM per la gestione del video. Sapendo che la risoluzione grafica è di 256* 192 punti, otteniamo con una semplice moltiplicazione il numero di bit necessari per la mappa completa dello schermo, ma una mappa di questo tipo sarebbe in bianco e nero, infatti un bit può assumere solo due valori: zero o uno, cioè punto bianco o punto nero. Alla cifra precedentemente ottenuta devono quindi essere aggiunti altri 768 bytes (badate bene, bytes, non bit). Perchè proprio 768? È semplice, 768 è infatti proprio il numero di posizioni-carattere che lo schermo dello Spectrum può contenere e, dato che in ogni posizione-carattere non si possono mischiare più di due colori, a luminosità normale o extra, lampeggiante o meno, si ottiene, considerando che si hanno a disposizione 8 colori, che un byte può contenere proprio le informazioni necessarie per gli attributi (colori, luminosità, ecc.) di una posizione-carattere dello schermo. Vediamo come.

Ogni byte è composto da 8 bit, e può quindi rappresentare un massimo di 256 valori. Nel nostro caso ci occorrerà un bit che indichi se il carattere è fisso (Ø) o lampeggiante (1), e un altro che indichi se è a luminosità normale (Ø) o extra (1), rimangono quindi disponibili 6 bit; come sappiamo bene 3 bit possono rappresentare fino a 8 valori, ecco quindi che di questi rimanenti 6 bit, tre servono a indicare il colore dello sfondo di quel singolo carattere (Ø-7), e tre indicano invece il colore dell'inchiostro (Ø-7). Come vedete con questo sistema si ottimizza il consumo di memoria, senza sacrificare la grafica.

Tornando a quanto detto precedentemente, si ricava che la memoria richiesta dallo Spectrum per la gestione completa del display file, ammonta a:

(256*192)/8 + 768 =
= 6912 bytes

sapendo che il display file inizia alla locazione 16384, ne ricaviamo che esso termina all'indirizzo 23295; sottraendo da quest'ultimo valore la cifra 768, otteniamo l'area occupata dagli attributi che va appunto dalla locazione 22528 alla 23295. Sapendo questo, risulta molto semplice cambiare a piacere gli attributi di una determinata posizione dello schermo con un semplice POKE nella locazione corrispondente. Il valore da attribuire a una determinata posizione si ottiene con un semplice calcolo che, con l'abitudine, diviene ben presto intuitivo (il valore così calcolato è lo stesso che si ottiene come risultato applicando la funzione ATTR a una posizione dello schermo). Il procedimento è il seguente:

128*1 se lampeggiante o Ø se fissa +
64*1 se luminosità extra o Ø se non +
8*(Ø-7) colore di sfondo +
(Ø-7) colore per l'inchiostro

Quindi se noi usiamo la funzione ATTR all'accensione del computer, essa riporterà, in qualunque posizione la si applichi, il valore 56. Infatti la luminosità è a zero (Ø*64 = Ø), il lampeggio anche (Ø*128 = Ø), lo sfondo è ovunque bianco (8*7 = 56), e l'inchiostro è nero (Ø), quindi Ø + Ø + 56 + Ø = 56. Semplice vero?

Da quanto è stato detto si può facilmente comprendere come con un semplice programma che esegue una serie di POKE nell'area degli attributi sia possibile cambiare questi ultimi lasciando invariato quanto si tro-

va sullo schermo. Ciò può essere utile, per esempio, nel caso in cui si debba realizzare un disegno sullo schermo che richiede un certo tempo per essere eseguito; è infatti possibile realizzare prima il disegno con un valore di INK uguale a quello di sfondo, in questo modo non si vedrà il disegno formarsi, e subito dopo eseguire una routine di questo genere:

```
1∅ FOR A = 22528 TO 23295
2∅ POKE A,4∅
3∅ NEXT A
```

con la quale si ottiene uno schermo azzurro con inchiostro nero.
Purtroppo una routine del genere impiega alcuni secondi per essere eseguita, privando di parte del fascino questo bell'effetto. Per questo motivo riportiamo qui di seguito l'equivalente programma in linguaggio macchina e, per chi lo conosce, il listato assembler.

```
1∅ REM CARICAMENTO
   CODICE MACCHINA
2∅ CLEAR 31999
3∅ FOR A = ∅ TO 13
35 READ N
4∅ POKE 32∅∅∅ + A,N
5∅ NEXT A
9∅ DATA 33,∅,88,17,1,88,1,
   255,2,54,4∅,237,176,2∅1
```

Dopo aver fatto girare il programma con un RUN, potete tranquillamente cancellarlo con NEW, in quanto il codice macchina è stato caricato sopra la ramtop e, di conseguenza, non viene influenzato da questo comando. Per mandare in esecuzione la routine usate RANDOMIZE USR 32∅∅∅.

Per salvarla su nastro usate SAVE "nome" CODE 32∅∅∅,14 mentre per ricaricarla in una zona a piacere della memoria (di solito il più in alto possibile) dovrete usare LOAD "nome" CODE xxx: CLEAR xxx-1 e per farla eseguire un RANDOMIZE USR xxx, dove xxx è la locazione di inizio della routine. Naturalmente potete fare assumere allo schermo qualunque colorazione semplicemente modificando l'undicesimo byte con un POKE 32∅1∅,x (a condizione che abbiate usato come locazione di partenza la 32∅∅∅), dove x rappresenta un qualsiasi numero da ∅ a 255 calcolato nel modo sopra descritto. Questa routine impiega circa 4 millesimi di secondo per essere eseguita e risulta quindi istantanea. Per i più esperti ecco il corrispondente listato assembler.

```
LD HL,22528
LD DE,22529
LD BC,767
LD (HL),4∅
LDIR
RET
```

Un programma di questo genere può essere molto utile anche per trasferire degli schermi in altre parti della memoria, e richiamarli successivamente, ma di questo parleremo la prossima volta.
Per il momento concludiamo la nostra esplorazione tra i meandri dello Spectrum, ma prima di lasciarvi, un consiglio: se non siete sicuri di aver capito bene quanto esposto in questo articolo fate delle prove, modificate magari i programmi che vi proponiamo (non quello con il codice macchina!!!), sperimentate finchè non siete sicuri di aver capito bene questi argomenti, e ricordatevi che in certi casi dieci minuti di pratica valgono quanto un ora di teoria.

Massimo Cellini

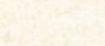
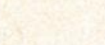
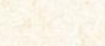
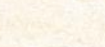
FROGWAY

In attesa del prossimo numero vi proponiamo il listato di un programma che si ispira a un famoso gioco arcade: FROGGER.

Benché si tratti di un programma interamente in BASIC, il gioco risulta piacevole, grazie anche alla grafica particolarmente curata.

Riteniamo che sarebbe un buon esercizio cercare di capire come funziona questo breve programma. In fondo con qualcosa bisogna pur cominciare...

```
  10 BORDER 1: PAPER 1: BRIGHT 1
: INK 7: CLS
  25 PRINT AT 10,13;"FROGWAY"
  35 PRINT AT 21,1; FLASH 1;"pre
mi un tasto per continuare"
  45 PAUSE 0
  55 DIM D(8): DIM L$(8,60): DIM
 I(21): DIM P(21)
  65 GO SUB 405
  75 LET V=2: PRINT AT 0,25; INK
 3; PAPER 6;"VITE:2"
  85 LET P=0
  95 LET T=0
 105 LET X=16: LET Y=21: LET C$=
" "

 115 IF P(Y)=7 THEN LET I(Y)=0
 125 PRINT AT Y,X; INK I(Y); PAP
ER P(Y);"X": LET AX=X: LET AY=Y
 135 LET A=1
 145 FOR I=1 TO 8
 155 PRINT AT I*2+A,0; INK I(I*2
+A); PAPER P(I*2+A);L$(I,D(I) TO
 D(I)+31)
 165 IF I=4 THEN LET A=3
 175 NEXT I
 185 IF Y<=3 THEN GO TO 795
 195 FOR I=1 TO 7 STEP 2
 205 LET D(I)=D(I)+1
 215 IF D(I)=25 THEN LET D(I)=1
 225 LET D(I+1)=D(I+1)-1
 235 IF D(I+1)=0 THEN LET D(I+1)
=24
 245 NEXT I
 255 PRINT AT 0,5; INK 3; PAPER
6;"PUNTI: ";P
```

```
 265 LET A$=INKEY$
 275 IF A$="x" THEN LET X=X+1
 285 IF A$="z" THEN LET X=X-1
 295 IF A$=" " THEN LET Y=Y-2: L
ET P=P+1: BEEP .01,10
 305 IF AY=11 OR AY=21 THEN PRIN
T AT AY,AX; INK I(AY); PAPER P(A
Y);" "
 315 LET C$=SCREEN$ (Y,X)
 325 IF Y<11 AND C$=" " THEN GO
TO 725
 335 IF Y>12 AND C$="" THEN GO T
O 725
 345 IF X<1 OR X>30 THEN GO TO 7
25
 355 IF AY<>Y THEN GO TO 125
 365 IF Y=3 OR Y=7 THEN LET X=X-
1
 375 IF Y=5 OR Y=9 THEN LET X=X+
1
 385 GO TO 125
 395 STOP
 405 FOR I=0 TO 135
 415 READ A: POKE USR "A"+I,A
 425 NEXT I
 435 DATA 63,72,136,255,128,255,
136,112,224,16,8,255,1,255,34,28
 445 DATA 7,8,16,255,128,255,68,
56,248,36,34,255,1,255,17,14
 455 DATA 254,142,142,254,255,25
5,68,56,127,113,113,127,255,255,
34,28
 465 DATA 15,23,35,65,65,35,23,1
5,240,248,252,254,254,252,248,24
0
 475 DATA 240,232,196,130,130,19
6,232,240,15,31,63,127,127,63,31
,15
 485 DATA 28,73,127,28,28,62,99,
99,255,255,255,254,255,255,255,2
55
 495 DATA 255,255,255,255,255,25
5,68,56,255,255,255,255,255,255,
0,0
 505 DATA 0,192,96,48,31,13,7,25
5,0,0,0,0,255,255,255,255,0,0,0,
0,192,240,252,255
 515 LET L$(1)="
"
 525 LET L$(2)="
"
 535 LET L$(3)=L$(1)
 545 LET L$(4)=L$(2)
 555 LET L$(5)="
```

CARATTERI GRAFICI:

ISTRUZIONI PER LA CASSETTA DI COMPUTING VIDEOTECA N. 2

COME INIZIARE

Per caricare i programmi contenuti nella cassetta battere: LOAD "" (le " si ottengono con simbolo SHIFT e P) dopodiché avviare il registratore e seguire le indicazioni che appariranno sul video.
Quando avrete finito di usare un programma fermatelo con BREAK (CAPS SHIFT e SPACE) e battete di nuovo LOAD "" per caricare il successivo.

FROG-RACE

Questa volta vi proponiamo un programma che sarà senz'altro gradito da quanti amano le scommesse e, in particolare dai più accaniti frequentatori degli ippodromi.
FROG-RACE non è però una comune corsa di cavalli, ma bensì una avvincente corsa fra rane!!!
Delle cinque rane che gareggiano voi dovete sceglierne una su cui puntare i vostri soldi. Le vincite saranno pagate 5 a 1.
Datevi da fare!

TORRE LASER

In questo gioco siete il comandante dell'ultima torre laser rimasta a difesa della Terra ormai completamente in balia dei mostruosi alieni che, a bordo dei loro veloci dischi volanti, cercano in tutti i modi di distruggervi.
Dovete assolutamente distruggerli prima che vi raggiungano o sarà la fine...
Ma voi resisterete per non permettere che la Terra cada in mani nemiche e anche perché ogni astronave distrutta vi frutterà ben 20 punti.
CORAGGIO!

PAC-CHIAN

Ormai i videogiocatori diventano sempre più esigenti, e i poveri (sic) programmatori di videogames devono sempre cercare nuove idee per attirare l'attenzione dei nuovi killer del joystick.

Questo originale gioco che vi proponiamo può giustamente essere considerato una divertente parodia di PAC-MAN, con immensa gioia di quanti soffrono ancora di PAC-ALLERGIA o malattie derivate.

Lo scopo del gioco consiste nel pitturare lo schermo di blu, ma naturalmente ci sono gli immancabili monelli che lo risporcano di rosso.

Per fermare questi monelli occorre usare la barriera (si ottiene premendo i tasti di direzione insieme a ∅), per mezzo della quale si può intrappolarli per far sì che non disturbino. Il punteggio dipende dall'area colorata e dal tempo risparmiato; esiste infatti anche un tempo limite entro cui terminare l'impresa.

Buon divertimento...

DIETA

Anche se ormai l'estate stà per finire, bisogna comunque pensare a mantenersi in forma.

Un valido aiuto in questo arduo compito ci viene fornito dal nostro fido spectrum il quale grazie a questo programma, vi fornirà sempre i giusti consigli per mantenere una linea invidiabile.

Il programma presenta due opzioni principali:

calcolo calorie

tabella alimenti

Il calcolo delle calorie vi fornirà, in base ai dati da voi introdotti, il giusto numero di calorie per conservare o raggiungere il peso forma.

La tabella alimenti visualizza invece le caratteristiche principali di molti degli alimenti più comuni e si divide a sua volta in 15 categorie di cibi.

Buona dieta dunque! E non prendetevela con il vostro computer se non otterrete risultati, lui non deve sopportare penosi digiuni...

Numeri già apparsi:

VIDEOTECA COMPUTER N. 1
per i possessori di Commodore 64

Nel manuale n. 1: I tasti funzionali del Commodore 64 - Pseudocodice e programmazione Basic - Il joystick.
Nella cassetta n. 1: Slalom - Slot Machine - Bilancio familiare - Briscola - Domino.

PLAY ON TAPE N. 1
per i possessori di VIC 20

Nel manuale n. 1: I tasti funzionali del VIC 20 - Pseudocodice e programmazione Basic - Il joystick.
Nella cassetta n. 1: Totocalcio - Air attack - Cervellone - Inferno 3D - Bilancio familiare.

VIDEOTECA COMPUTER N. 2
per i possessori di Commodore 64

Nel manuale n. 2: Il basic più veloce - Una migliore gestione del video per il 64 - Disegnare con tastiera e joystick - Pseudocodice: 2ª lezione.
Nella cassetta n. 2: Tennis 3d - Totocalcio - Gestione magazzino - Wargame - Colour search.

COMPUTING VIDEOTECA N. 1
per i possessori di Sinclair SPECTRUM ZX

Nel manuale n. 1: Pseudocodice e programmazione basic: 1ª e 2ª lezione - La gestione dei canali dello Spectrum - Come farsi una cassetta di "Subroutines".
Nella cassetta: Wargame - Gestione del magazzino - U.F.O. - Helibomber.

I numeri arretrati costano L. 15.000. Indirizzare vaglia o assegno a Editoriale VIDEO via Castelvetro 9 - 20154 Milano specificando il numero richiesto.
Ufficio tecnico e arretrati: telefono 02/3184829

PLAY ON TAPE N. 2

Nel manuale n. 2: Il basic più veloce - I caratteri speciali del VIC 20 - Disegnare con la tastiera e il joystick - Pseudocodice: 2ª lezione.
Nella cassetta n. 2: Test per misurare il Quoziente intellettuale - Easyword - Caccia al tesoro - Gestione Magazzino - Formula 1.

VIDEOTECA COMPUTER N. 3
per i possessori di Commodore 64

Nel manuale n. 3: I cicli annidati del Basic - Come sviluppare un programma (Squash e Trampolino) - Conosci il tuo CBM 64?
Nella cassetta n. 3: Starway - Easyword - Poker - Forza 4 - Test Quoziente Intellettuale.

Editoriale VIDEO

COMPUTING VIDEOTECA n. 2

Direttore: Antonio Lucarella
Coordinamento tecnico:
Roberto Treppiedi

Hanno collaborato:
Alberto Barbati
Massimo Cellini
Galeazzo Conti
Umberto Colapicchioni
Cinzia Cimbali
Umberto Fassi
Alfredo Malgrati
Stefano Milanesi
Maurizio Monteverdi
Sebastiano Pastore
Glauco Piatteletti
Alessandro Vallone
Giovanna Zampella

Stampa:
Tipolito FE.ZA. Milano
Fotocomposizione:
ERREGI Milano

Stampato a Milano

manuale 2

Per i possessori di computer SINCLAIR ZX SPECTRUM

PSEUDOCODICE: 3ª lezione

Come sviluppare un programma

Variabili interne e gestione del video